Solutions

Each answer shows the steps, then the **result**.

2.1

- binary → denary: multiply each 1-bit by 2^n (position n counts from 0 at the right)
- 1-bits at positions $6, 5, 3, 1$: $2^6 + 2^5 + 2^3 + 2^1$
- $64 + 32 + 8 + 2 =$ **106**

2.2

- denary → binary: divide by 2, record remainders, read **bottom to top**

```
200 ÷ 2 = 100  rem 0
100 ÷ 2 =  50  rem 0
 50 ÷ 2 =  25  rem 0
 25 ÷ 2 =  12  rem 1
 12 ÷ 2 =   6  rem 0
  6 ÷ 2 =   3  rem 0
  3 ÷ 2 =   1  rem 1
  1 ÷ 2 =   0  rem 1   ← left-most bit
```

- remainders bottom → top: `11001000`
- check: $128 + 64 + 8 = 200$

2.3

(a) binary → denary

- 1-bits at positions $7, 6, 5, 4, 0$: $128 + 64 + 32 + 16 + 1 =$ **241**

(b) binary → hex: split into nibbles from the right

- `1111` = 15 = F and `0001` = 1, so **F1**

2.4

- need the smallest b with $2^b > 500$
- $2^8 = 256$ is too small; $2^9 = 512 > 500$
- **9 bits**

2.5

- BCD stores **each denary digit** as its own 4-bit code
- 5 → `0101`, 9 → `1001`
- 59 in BCD = `0101 1001`

2.6

- add one column at a time from the right

```
  0101 1100   (92)
+ 0011 0110   (54)
------------
  1001 0010   (146)
```

- $146 \leq 255$ (no carry out of bit 7), so **no overflow**

3.1

- add the columns from the right

```
  1011 0100   (180)
+ 0110 1101   (109)
------------
1 0010 0001   (carry out of bit 7)
```

- stored 8-bit result = `00100001`
- $180 + 109 = 289 > 255$, so a 1 is lost from bit 7
- the error is **overflow**: the result needs 9 bits but only 8 are stored, so the stored answer is wrong

3.2

(a) hex → denary: powers of 16

- `D` = 13
- $3 \times 16^1 + 13 \times 16^0 = 48 + 13 =$ **61**

(b) denary → hex: divide by 16

- $158 \div 16 = 9$ remainder 14
- remainder 14 = E, so **9E**

3.3

(a) write −45 in 8-bit two's complement

- $+45 = 32 + 8 + 4 + 1$ → `00101101`
- invert every bit → `11010010`
- add 1 → `11010011`

(b) read `11101110`

- sign bit is 1, so the value is **negative**
- invert → `00010001`
- add 1 → `00010010` = 18
- denary value = **−18**

3.4

(a) any **two** of:

- Unicode can represent characters from (almost) every language / far more characters than ASCII
- it includes symbols and emoji
- it avoids the regional clashes of extended ASCII

(b) binary → denary

- 1-bits at positions $5, 2, 1, 0$: $32 + 4 + 2 + 1 = \mathbf{39}$

3.5

(a) **Binary Coded Decimal**

(b) any **one** of:

- each denary digit is stored exactly, so money values are not changed by rounding errors when converting fractions to binary
- BCD digits map directly onto a seven-segment display